CS 4444/6444 Spring 2017
Assignment 4: Heated Plate with Pthreads
Due by 5:00 PM on Tuesday, April 13

Introduction

Your task in this assignment is to implement a parallel version of the heated plate problem using Pthreads. You will run your parallel program on the Hermes nodes in the CS department. If you do not have a CS department account please send root@cs.virginia.edu email requesting an account AND BE SURE TO CC me. When running you want to make sure that you acquire the entire node using SLURM or your performance results will be skewed.

Files needed

To start, download from the course Collab site the following files:

•heated_plate.c: this is the file that you will need to parallelize.

• ground_truth.ppm: a snapshot generated by the sequential implementation of a 10,000 x 10,000 heated plate after 10,000 iterations. Use this to verify that your parallel implementation is producing the correct results.

• create_jpegs.pl: run this Perl script to convert PPM snapshots of the heated plate into JPEG images.

The program

The program in this assignment models the flow of heat through a two-dimensional plate. The plate is divided up into a grid of cells, and the temperature of each cell is computed in discrete time steps. The interior cells are all initialized to the same temperature (50 degrees), and the border cells are fixed at a specific temperature (0 degrees along the top and left sides and 100 degrees along the bottom and right sides). In each time step of the simulation, the temperature of each cell is computed by averaging the temperatures of the four neighboring cells in the previous time step.

To spice it up this year we will add a single internal boundary condition in which a single cell is held at 1000 degrees. The single cell is at position row=4500, column = 6500. If those coordinates are outside the specified plate size then it is ignored.

The original version of the program provided to you is sequential and not optimized; all of the computation is performed by a single thread. After simulating a specified number of iterations, the final state of the plate is output as an image in the PPM format. You can use the script provided (create_jpegs.pl) to convert the PPM image(s) into JPEG format for easier viewing. For fun you can generate a sequence of images by periodically creating an image, and then create a movie.

To run the program, specify the size of the plate and the number of iterations as follows:

```
./heated_plate <columns> <rows> <iterations>
```

For example, to simulate a 500 x 500 plate for 1,000 iterations, use the following command:

```
./heated_plate 500 500 1000
```

The command-line arguments are optional; default values will be used if they are missing.

The task

The task in this assignment is to parallelize the sequential version of the program using Pthreads. The number of threads used by your program should be controllable via a command-line argument. Please use the snapshot output from the sequential implementation to verify that your parallel implementation is executing correctly. We will provide snapshot output for a 10,000 x 10,000 plate after 10,000 iterations (ground_truth.ppm) so you can verify correct execution of your final runs. For testing of your initial (shorter) runs, you can use the sequential implementation to generate your own PPM files to compare against. Once you have a working Pthreads implementation, measure the performance impact of increasing the number of threads (and the number of processor cores). You should measure performance for thread counts of all powers of 2 from 1 through 64 (1, 2, 4, 8, 16, 32, and 64). There are 64 cores on each Hermes node.

When measuring the final execution times that you will provide in your report, use the following parameter values: a grid size of 10,000 x 10,000 and 10,000 iterations. For initial testing, you will most likely want to use a smaller number of iterations.

NOTE: Make sure that you always ask for exclusive access to the node from SLURM or your performance results will be difficult to understand or replicate.

Optimization

The code is not optimized for modern shared memory machines. If you simply slap some threads on this code it will not perform particularly well, and your grade will reflect that. To receive full points on this homework you must also optimized your code for the target machine AND CLEARLY IDENTIFY THOSE OPTIMIZATIONS AND THE EFFECT THEY HAD ON PERFORMANCE. Hint. Memory is not flat.

Homework point breakdown:

50 points for the write up

50 points for code

- 10 points for comments
- 10 points for correct threaded implementation
- 30 points for performance optimizations